AMENDMENTS TO THE CLAIMS

1. (Currently Amended) A method comprising:

identifying a plurality of fork subgraph structures within a graph structure constructed for a plurality of ~~executable~~ source code instructions;

identifying, prior to register allocation, a plurality of unifiable variables within each fork subgraph structure of said plurality of fork subgraph structures, which are not simultaneously used in said plurality of ~~executable~~ source code instructions;

constructing a dependence graph of said plurality of ~~executable~~ source code instructions;

using said dependence graph to identify at least one unifiable instruction of said plurality of executable instructions, within said plurality of fork subgraph structures; ~~and~~

transferring said at least one unifiable instruction of said plurality of executable instructions from a tine of a corresponding fork subgraph structure of said plurality of fork subgraph structures to a handle of said corresponding fork subgraph structure, said at least one unifiable instruction containing at least one unifiable variable of said plurality of unifiable variables~~.~~; and

unifying each unifiable variable within said at least one unifiable instruction.

2. (Previously Presented) The method according to claim 1, wherein identifying said plurality of unifiable variables further comprises:

constructing an interference graph structure for a plurality of local variables within said each fork subgraph structure, said plurality of local variables including said plurality of unifiable variables; and

identifying said plurality of unifiable variables as variables having non-overlapping live ranges within said interference graph structure.

3. (Currently Amended) The method according to claim 2, wherein said interference graph structure indicates which variables of said plurality of local variables are simultaneously used in said plurality of ~~executable~~ source code instructions and cannot be unified.

4. (Currently Amended) The method according to claim 1, wherein identifying said plurality of unifiable variables further comprises:

constructing a data dependence graph structure for said plurality of ~~executable~~ source code instructions; and

identifying said plurality of unifiable variables using said data dependence graph structure.

5. (Currently Amended) The method according to claim 1, wherein said transferring further comprises:

initializing a flag for said at least one unifiable instruction~~; and~~.

~~unifying each unifiable variable within said at least one unifiable instruction.~~

6. (Previously Presented) The method according to claim 5, wherein said transferring further comprises:

removing said at least one unifiable instruction from subsequent tines of said corresponding fork subgraph structure.

7. (Currently Amended) The method according to claim 4, wherein said data dependence graph structure contains a plurality of dependence arcs, each dependence arc connecting two instructions of said plurality of ~~executable~~ source code instructions contained within said tine of said corresponding fork subgraph structure.

8. (Currently Amended) A system comprising:

means for identifying a plurality of fork subgraph structures within a graph structure constructed for a plurality of ~~executable~~ source code instructions;

means for identifying, prior to register allocation, a plurality of unifiable variables within each fork subgraph structure of said plurality of fork subgraph structures, which are not simultaneously used in said plurality of ~~executable~~ source code instructions;

means for constructing a dependence graph of said plurality of ~~executable~~ source code instructions;

means for using said dependence graph to identify at least one unifiable instruction of said plurality of ~~executable~~ source code instructions, within said plurality of fork subgraph structures; ~~and~~

means for transferring said at least one unifiable instruction of said plurality of ~~executable~~ source code instructions from a tine of a corresponding fork subgraph structure of said plurality of fork subgraph structures to a handle of said corresponding fork subgraph structure, said at least one unifiable instruction containing at least one unifiable variable of said plurality of unifiable variables~~.~~; and

means for unifying each unifiable variable within said at least one unifiable instruction.

9. (Previously Presented) The system according to claim 8, further comprising:

means for constructing an interference graph structure for a plurality of local variables within said each fork subgraph structure, said plurality of local variables including said plurality of unifiable variables; and

means for identifying, prior to register allocation, said plurality of unifiable variables as variables having overlapping live ranges within said interference graph structure.

10. (Currently Amended) The system according to claim 9, wherein said interference graph structure indicates which variables of said plurality of local variables are simultaneously used in said plurality of ~~executable~~ source code instructions and cannot be unified.

11. (Currently Amended) The system according to claim 8, further comprising:

means for constructing a data dependence graph structure for said plurality of ~~executable~~ source code instructions; and

means for identifying said plurality of unifiable variables using said data dependence graph structure.

12. (Currently Amended) The system according to claim 8, further comprising:

means for initializing a flag for said at least one unifiable instruction; and

~~means for unifying each unifiable variable within said at least one unifiable instruction.~~

13. (Previously Presented) The system according to claim 12, further comprising:

means for removing said at least one unifiable instruction from subsequent tines of said corresponding fork subgraph structure.

14. (Currently Amended) The system according to claim 11, wherein said data dependence graph structure contains a plurality of dependence arcs, each dependence arc connecting two instructions of said plurality of ~~executable~~ source code instructions contained within said tine of said corresponding fork subgraph structure.

15. (Currently Amended) A computer readable medium containing instructions, which, when executed in a processing system, cause said processing system to perform a method comprising:

identifying, prior to register allocation, a plurality of fork subgraph structures within a graph structure constructed for a plurality of ~~executable~~ source code instructions;

identifying a plurality of unifiable variables within each fork subgraph structure of said plurality of fork subgraph structures, which are not simultaneously used in said plurality of ~~executable~~ source code instructions;

constructing a dependence graph of said plurality of ~~executable~~ source code instructions;

using said dependence graph to identify at least one unifiable instruction of said plurality of ~~executable~~ source code instructions, within said plurality of fork subgraph structures; ~~and~~

transferring said at least one unifiable instruction of said plurality of ~~executable~~ source code instructions from a tine of a corresponding fork subgraph structure of said plurality of fork subgraph structures to a handle of said corresponding fork subgraph structure, said at least one unifiable instruction containing at least one unifiable variable of said plurality of unifiable variables~~.~~; and

unifying each unifiable variable within said at least one unifiable instruction.

16. (Previously Presented) The computer readable medium according to claim 15, wherein identifying said plurality of unifiable variables further comprises:

constructing an interference graph structure for a plurality of local variables within said each fork subgraph structure, said plurality of local variables including said plurality of unifiable variables; and

identifying said plurality of unifiable variables as variables having overlapping live ranges within said interference graph structure.

17. (Currently Amended) The computer readable medium according to claim 16, wherein said interference graph structure indicates which variables of said plurality of local variables are simultaneously used in said plurality of ~~executable~~ source code instructions and cannot be unified.

18. (Currently Amended) The computer readable medium according to claim 15, wherein identifying said plurality of unifiable variables further comprises:

constructing a data dependence graph structure for said plurality of ~~executable~~ source code instructions; and

identifying said plurality of unifiable variables using said data dependence graph structure.

19. (Currently Amended) The computer readable medium according to claim 15, wherein said transferring further comprises:

initializing a flag for said at least one unifiable instruction~~; and~~.

~~unifying each unifiable variable within said at least one unifiable instruction.~~

20. (Previously Presented) The computer readable medium according to claim 19, wherein said transferring further comprises:

removing said at least one unifiable instruction from subsequent tines of said corresponding fork subgraph structure.

21. (Previously Presented) The computer readable medium according to claim 18, wherein said data dependence graph structure contains a plurality of dependence arcs, each dependence arc connecting two instructions of said plurality of executable instructions contained within said tine of said corresponding fork subgraph structure.

22. (Currently Amended) A system comprising:

a memory having stored thereon a plurality of executable instructions to be executed in a program; and

a processor coupled to said memory

to identify a plurality of fork subgraph structures within a graph structure constructed for said plurality of ~~executable~~ source code instructions;

to identify, prior to register allocation, a plurality of unifiable variables within each fork subgraph structure of said plurality of fork subgraph structures, which are not simultaneously used in said plurality of ~~executable~~ source code instructions;

to construct a dependence graph of said plurality of ~~executable~~ source code instructions;

to use said dependence graph to identify at least one unifiable instruction of said plurality of ~~executable~~ source code instructions, within said plurality of fork subgraph structures; ~~and~~

to transfer said at least one unifiable instruction of said plurality of ~~executable~~ source code instructions from a tine of a corresponding fork subgraph structure of said plurality of fork subgraph structures to a handle of said corresponding fork subgraph structure, said at least one unifiable instruction containing at least one unifiable variable of said plurality of unifiable variables~~.~~; and

to unify each unifiable variable within said at least one unifiable instruction.

23. (Previously Presented) The system according to claim 22, wherein said processor further constructs an interference graph structure for a plurality of local variables within said each fork subgraph structure, said plurality of local variables including said plurality of unifiable variables and further identifies said plurality of unifiable variables as variables having overlapping live ranges within said interference graph structure.

24. (Currently Amended) The system according to claim 23, wherein said interference graph structure indicates which variables of said plurality of local variables are simultaneously used in said plurality of ~~executable~~ source code instructions and cannot be unified.

25. (Original) The system according to claim 22, wherein said processor further constructs a data dependence graph structure for said plurality of executable instructions and identifies said plurality of unifiable variables using said data dependence graph structure.

26. (Original) The system according to claim 22, wherein said processor further initializes a flag for said at least one unifiable instruction and unifies each unifiable variable within said at least one unifiable instruction.

27. (Previously Presented) The system according to claim 26, wherein said processor further removes said at least one unifiable instruction from subsequent tines of said corresponding fork subgraph structure.

28. (Currently Amended) The method according to claim 25, wherein said data dependence graph structure contains a plurality of dependence arcs, each dependence arc connecting two instructions of said plurality of ~~executable~~ source code instructions contained within said tine of said corresponding fork subgraph structure.